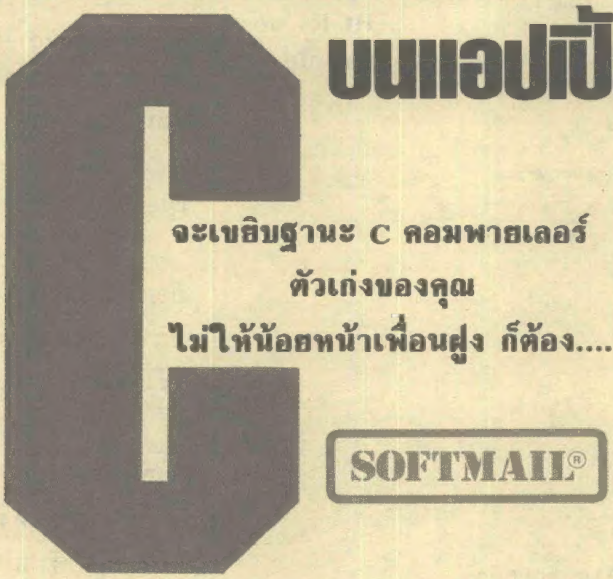
สร้างกราฟิกไลบรารี่ภาษา C บนแอปเปิ้ล



ศิระ ลิ้มเจริญ

ภาษา C นั้นเป็นภาษาที่เรียกได้ว่ามีประสิทธิภาพที่สุดในบรรดาภาษาคอมพิวเตอร์ที่เรารู้จักกัน ก่อนที่ผมจะเลือกใช้ภาษา C เป็นภาษาหลักของผมนั้น ผมก็เคยลองใช้มาแล้วหลายภาษา แต่ก็รู้สึกว่าภาษาเหล่านั้นยังขาดหลายๆ อย่างที่ภาษา C มี ไม่ว่าจะเป็นภาษา BASIC FORTRAN, FORTH, PASCAL หรือแม้แต่ ASSEMBLY ภาษา C นอกจากจะลงถึงฮาร์ดแวร์ของเครื่องได้อย่างชนิดที่เทียบเท่ากับภาษา ASSEMBLY ตัวต่อตัวแล้ว ยังแฝงลักษณะของภาษาระดับสูงเอาไว้ด้วย เคยมีผู้กล่าวไว้ว่าภาษา C นั้นสร้างตัวเองได้ สมมติว่าคุณมีคอมพายเลอร์แบบพื้นๆ อยู่ตัวหนึ่งซึ่งไม่มีฟังก์ชันอะไรมากมาย จากคอมพายเลอร์ตัวเล็กนี้แหละที่คุณสามารถใช้เขียน C คอมพายเลอร์ ตัวใหญ่ที่มีฟังก์ชันเท่ากับของ UNIX C ได้ นอกจากที่กล่าวมาแล้วสิ่งที่ภาษา C กินขาดภาษาอื่นก็คือความเป็นมาตรฐานของภาษา C เพราะ C คอมพายเลอร์ในเครื่องระดับไมโครคอมพิวเตอร์แทบทุกตัว จะสร้างฟังก์ชันในไลบรารี่ตาม UNIX C ทั้งสิ้น

ด้วยเหตุนี้เองผมจึงสามารถเขียนภาษา C ใช้งานได้ทั้งบน แอปเปิ้ลทู และไอบีเอ็มพีซีในเวลาเดียวกัน บน ไอบีเอ็มนั้น C คอมพายเลอร์ ที่เก่งๆ ตอนนี้ก็มี Microsoft C Version 4.0 หรือถ้าตัวที่เก่าแก่แต่ยังใช้ได้ดีหน่อยก็คือ Lattice C ส่วนบนเครื่องแอปเปิ้ล ตัวคอมพายเลอร์ที่ผมใช้ก็คือ BDS C Compiler ถึงแม้เวอร์ชัน 1.4 จะดูเก่าไปสักหน่อยแต่มีให้ใช้งานบนเครื่อง 8 บิตสำหรับผู้ที่จะเริ่มต้นใช้ภาษา C

BDS C Compiler เป็นคอมพายเลอร์ของบริษัท BD Software ที่รันบน CP/M ส่วนตัวออบเจ็คโค้ดที่คอมพายล์แล้วจะใช้งานกับ CP/M หรือไม่ก็ได้แล้วแต่ว่าเราจะ Install Compiler ไว้อย่างไร BDS C มีฟังก์ชันในไลบรารี่เกือบ 100 ฟังก์ชัน ได้แก่ ฟังก์ชันทั่วไป, ฟังก์ชันอินพุต/เอาต์พุต, ฟังก์ชันที่เกี่ยวกับ String และ Character, File I/O และฟังก์ชันวาดรูปบนเท็กซ์ สิ่งที่คอมพายเลอร์ตัวนี้ทำไม่ได้คือ Long int, Float, Double, Static, Register และประโยคที่ซับซ้อนมากๆ จัดได้ว่าคอมพายเลอร์ตัวนี้เล็กพอสมควร ผู้เขียนคอมพายเลอร์ตัวนี้ได้บอกไว้ว่าเขาเขียนด้วย 8080 assembler language ไม่ได้เขียนด้วย C และเตือนว่าถ้าจะเขียนคอมพายเลอร์ด้วยตัว BDS C เองแล้วจะได้คอมพายเลอร์ที่ยาวมากกว่า 5 เท่า และรันช้ามากๆ

จากการที่ BDS C ไม่มีฟังก์ชันกราฟิก ผมจึงคิดว่าเราน่าจะลองเขียนฟังก์ชันกราฟิกง่ายๆ ไว้ใช้งานกันดูสักตั้ง เอาไว้เป็นกราฟิกไลบรารี่ของภาษา C ถ้าคุณยังไม่รู้จักภาษา C ละก็ลองเปิดอ่านเรื่องภาษา C จากไมโครฯเล่มเก่าๆ ดู แล้วมาเริ่มเขียนภาษา C กับผมเดี๋ยวนี้เลย ฟังก์ชันที่ผมจะแนะนำต่อไปนี้เป็นฟังก์ชันกราฟิกของแอปเปิ้ล

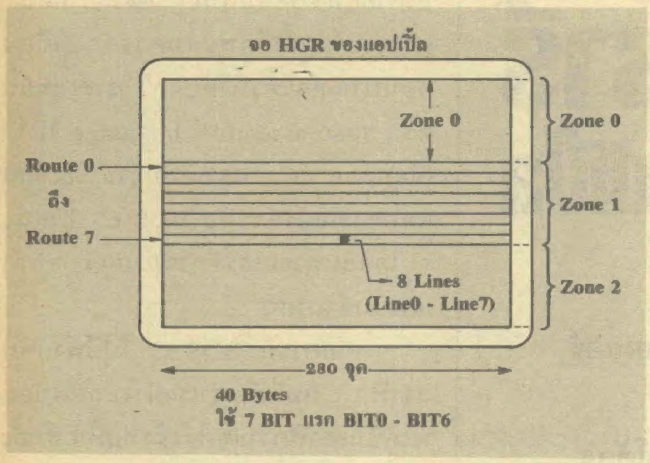
เริ่มต้นเรามาดูภาพกราฟิก HGR หน้า 1 กันก่อน

ซอฟต์สวิตช์สำหรับเปลี่ยนจอแอปเปิ้ลในขณะที่อยู่ใน CP/M มีดังนี้

1. Graphic screen (E050H)
2. Text screen (E051H)
3. Full screen (E052H)
4. Mixed screen (E053H)
5. Page 2 (E054H)
6. Page 1 (E055H)
7. Low resolution (E056H)
8. Hi-res. (E057H)

การใช้ซอฟต์สวิตช์ก็เพียงแต่อ้างอิงถึงแอดเดรสที่ต้องการเท่านั้น

จอ HGR ของแอปเปิ้ลเป็นแบบ Bit-map memory เริ่มจากแอดเดรส 3000h - 4FFFh ใน CP/M รวมทั้งสิ้น 8192 ไบต์ แต่ map ของแอปเปิ้ลไม่ได้เรียงกันง่ายๆ จากบนสุดของจอลงมาขอบล่างสุด นอกจากนั้นบางไบต์ก็ไม่ได้แสดงออกจอ ผมจึงสร้างสูตรสำหรับหาแอดเดรสของจุดที่ต้องการขึ้นมา โดยที่ผมได้แบ่งจอ HGR ออกเป็น 3 ส่วนใหญ่ๆ เรียกว่า Zone ใน 1 Zone จะประกอบด้วย 8 Routes และ แต่ละ Route จึง



รูปที่ 1 การตัดแบ่ง Zone บนจอ HGR ของแอปเปิ้ล

```
Hgr (Color)
char Color;
{ int i;
  char *Softsw,Swtch;
  Setmem(0x3000,8200,00);
  Softsw = 0xe050;
  Swtch = *(Softsw+ 0 ) ;/*  Graphic screen          */
  Swtch = *(Softsw+ 7 ) ;/*  Hires-Graphic Screen    */
  Swtch = *(Softsw+ 5 ) ;/*  Page 1                  */
  Swtch = *(Softsw+ 2 ) ;/*  Graphic Full Screen     */
}

Text ()
{ char *Softsw,Swtch;
  Softsw = 0xe050;
  Swtch = *(Softsw+ 1 ) ;/*  Text    screen          */
  Swtch = *(Softsw+ 4 )   ;/*  Page 1                  */
}

Hplot (Px,Py,Color)
int Px,Py ;
char Color;
{ char Hmask,*Hbase;
  if (Px >= 0)
     Px %= 280;
  else
     Px =(Px%280)+280;
  if (Py >= 0)
     Py %= 192;
  else
     Py =(Py%192)+192;
  Hbase = Hadrs(Px,Py);
  Hmask = Hbit(Px);
  *Hbase = (((*Hbase)^Color)&Hmask)^(*Hbase);
}

Hadrs (Px,Py)
int Px,Py;
{ int Zone,Line,Route,Adrs,Byt;
  Zone = Py>>6;
  Route = (Py&0x3f)>>3;
  Line = Py&0x07;
  Byt = Px/7;
  Adrs = (Line<<10)+(Route<<7)+(Zone*40)+Byt+0x3000;
  return (Adrs);
}

Hbit (Px)
int Px;
{ int Bit;
  char Mask;
  Bit = Px-((Px/7)*7);
  Mask = 0x01<<Bit;
  return (Mask);
}
```

รูปที่ 2 โปรแกรมฟังก์ชันพื้นฐานของกราฟิกไลบรารี่

จะเป็นเส้นที่พาดขวางจอ 8 Lines ดูรูปที่ 1 ประกอบเพื่อความเข้าใจ

สูตรต่อไปนี้ใช้ในการคำนวณหาแอดเดรสของไบต์ที่อยู่ชิดขอบซ้ายสุดของจอเมื่อทราบตำแหน่ง Line, Route และ Zone

Base address = (Line*2^{10}) + (Route*2^7) + (Zone*40)

เมื่อทราบแอดเดรสเริ่มต้นทางซ้ายมือแล้ว ต่อไปการหาแอดเดรสและตำแหน่งบิตของจุดที่ต้องการก็เป็นเรื่องง่ายแล้วใช่ไหมครับ

เรามาเริ่มดูโปรแกรมในรูปที่ 2 กันเลยนะครับ ตอนแรกผมจองตัวแปร Char เอาไว้ 8192 ตัว ซึ่งมีความยาวเท่ากับ 8192 ไบต์ ตัวแปรนี้เป็นตัวแปร External เวลาคอมพายล์จะต้องบอกตำแหน่งของตัวแปร External ด้วยว่าอยู่ที่ 3000h เพื่อกันหน่วยความจำส่วนที่เป็นจอกราฟิกเอาไว้ไม่ให้โดนใช้งานซ้อนกัน

ฟังก์ชันแรกคือ **Hgr(color)** เป็นฟังก์ชันสำหรับสวิตช์จอภาพจากเท็กซ์โหมดให้เป็น HGR หน้า 1 แบบเต็มจอ โดยมี color หมายถึง สีแบคกราวนด์ของจอภาพ ในที่นี้ผมเขียนฟังก์ชันกราฟิกขึ้นมาใช้งานกับจอมอนิเตอร์สีเขียวเท่านั้น สีของแบคกราวด์จึงมีแค่สองสีได้แก่สีขาว (สว่าง) และสีดำ (มืด)

เมื่อมีฟังก์ชัน Hgr แล้วก็ต้องมีฟังก์ชัน Text( ) สำหรับสวิตช์จอภาพกลับไปยังเท็กซ์โหมดตามเดิม

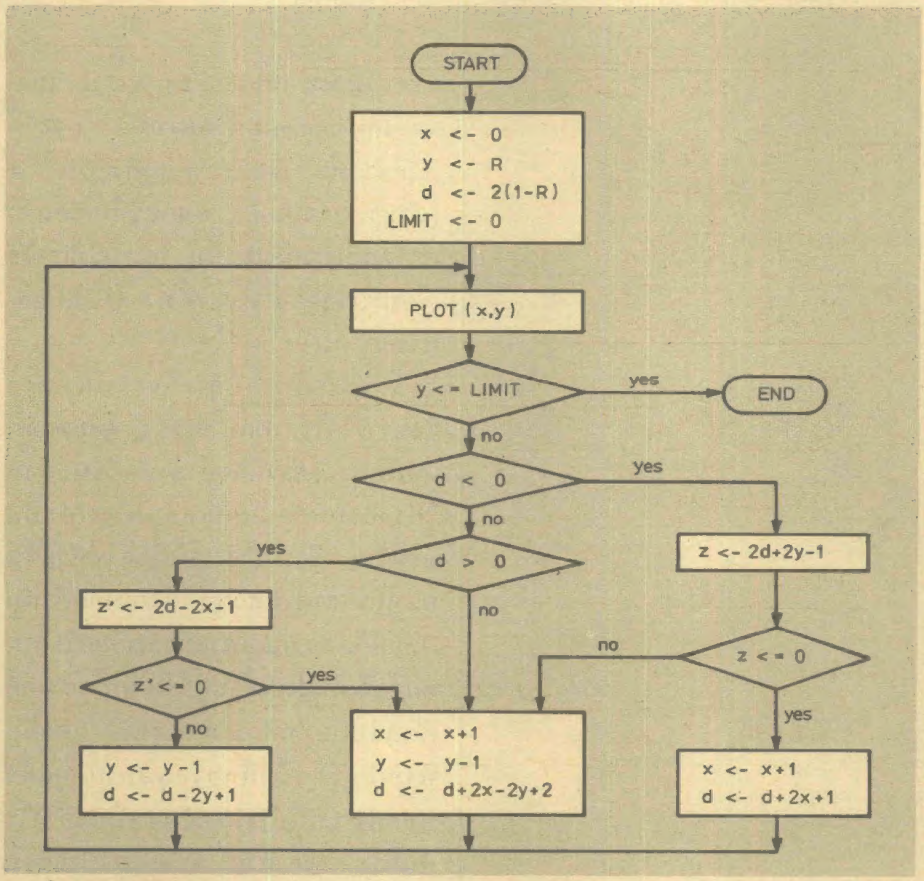
ฟังก์ชันต่อมาคือ **Hplot(Px, Py, Color)** ซึ่งจะ Plot จุดสีลงบนตำแหน่ง Px, Py ที่มี Origin อยู่ที่มุมซ้ายบนของจอ และเช่นเดียวกัน สี Color ในที่นี้หมายถึงสีดำหรือสีขาวเท่านั้น แต่ถ้าคุณต้องการให้เป็นสีอื่นจะต้องคำนวณตำแหน่งของจุดสีบนจอภาพใหม่

ฟังก์ชัน Hplot มีการเรียกใช้ฟังก์ชันอีกสองฟังก์ชันคือ **Hadrs(Px, Py)** และ **Hbit(Px)** ฟังก์ชัน Hadrs เอาไว้ใช้

คำนวณหาแอดเดรสของจุดบนจอภาพที่ต้องการ Plot โดยใช้สูตรที่ผมให้ไว้ข้างต้น ส่วน Hbit จะคำนวณหาตำแหน่งบิตเพื่อทำเป็น Mask ให้กับแอดเดรสที่เราหาได้ ฟังก์ชัน Hadrs และ Hbit จะทำให้ Hplot สามารถ plot จุดลงตำแหน่งที่ต้องการ

เพียงแค่ 3 ฟังก์ชัน คือ Hgr, Text และ Hplot คุณก็จะได้ฟังก์ชันกราฟิกง่ายๆ ไว้ใช้งานบน BDS C แล้ว ต่อไปผมก็จะนำฟังก์ชันกราฟิกที่เขียนขึ้นมานี้มาใช้งานในการวาดรูปวงกลม และ เส้นตรง สำหรับเก็บไว้เป็นกราฟิกไลบรารี่

ในการเขียนเส้นตรงและวงกลมนั้น มีอัลกอริทึมที่ใช้กันอย่างแพร่หลายอยู่แล้ว นั่นก็คืออัลกอริทึมของ Bresenham ผมจะให้รายละเอียดอัลกอริทึมและโฟลว์ชาร์ตในรูปที่ 3 เอาไว้ด้วยเผื่อคุณจะเอาไปดัดแปลงใช้กับภาษาอื่นได้อีก



รูปที่ 3 Flowchart ของ Bresenham's algorithm เขียนวงกลม

Generalized integer Bresenham's algorithm

เส้นตรงลากจากจุด (x1, y1) ไปยัง (x2 y2)

ตัวแปรทุกตัวเป็น integer

ฟังก์ชัน Sign ให้ค่า −1,0,1 เมื่อตัวแปรมีค่า < 0, = 0, หรือ > 0 ตามลำดับ

ตั้งค่าตัวแปร

```
x = x1
y = y1
dx = abs(x2 − x1)
dy = abs(y2 − y1)
s1 = Sign(x2 − x1)
s2 = Sign(y2 − y1)
```

สลับค่า dx กับ dy ตามความชันของเส้นตรง

```
if dy > dx then
    Temp = dx
    dx = dy
    dy = Temp
    Interchange = 1
else
    Interchange = 0
endif
```

ตั้งค่า error term

```
e = 2*dy − dx
main loop
for i = 1 to dx
    Plot(x, y)
    while (e > = 0)
      If Interchange = 1 then
        x = x + s1
      else
        y = y + s2
      endif
        e = e − 2*dx
      endwhile
      if Interchange = 1 then
        y = y + s2
      else
        x = x + s1
      endif
        e = e + 2*dy
next i
finish
```

Bresenham's incremental circle algorithm สำหรับ quadrant ที่ 1 ตัวแปรทุกตัวเป็น integer

ตั้งค่าตัวแปร

$x_i = 0$

$y_i = R$

$d_i = 2(1-R)$

Limit = 0

1 Plot (x_i, y_i)

if $y_i <=$ Limit then 4

ตัดสินใจกรณีที่ 1 หรือ 2, 4 หรือ 5, หรือ 3

if $d_i < 0$ then 2

```
circle(cx,cy,r,color)
int r,cx,cy;
char color;
{  int x,y,delta,sigma,limit;
   x=0; y=r; limit=0;
   delta = (1-r)*2;
   while (y >= limit ) {
      Hplot(cx+x,cy+y,color);
      Hplot(cx+x,cy-y,color);
      Hplot(cx-x,cy+y,color);
      Hplot(cx-x,cy-y,color);
      if (delta<0) {
         sigma = (2*delta)+(2*y)-1;
         if (sigma<=0)
            delta += 2*(++x)+1;
         else
            delta += 2*(++x)-2*(--y)+2;
      }
      else if (delta>0) {
         sigma = (2*delta)-(2*x)-1;
         if (sigma<=0)
            delta += 2*(++x)-2*(--y)+2;
         else
            delta -= 2*(--y)+1;
      }
      else if (delta==0)
         delta += 2*(++x)-2*(--y)+2;
   }
}

line(x1,y1,x2,y2,color)
int x1,y1,x2,y2;
char color;
{  int abs(),sign();
   int x,y,delx,dely,s1,s2,change,weight,temp,i;
   x = x1 ; y = y1;
   delx = abs(x2-x1);
   dely = abs(y2-y1);
   s1 = sign(x2-x1);
   s2 = sign(y2-y1);
   if (dely > delx) {
      temp = delx;
      delx = dely;
      dely = temp;
      change = 1;
   }
   else
      change = 0;
   weight = 2*dely - delx;
   for (i=1;i<=delx;i++) {
      Hplot(x,y,color);
      while (weight >= 0) {
         if (change == 1)
            x += s1;
         else
            y += s2;
         weight -= 2*delx;
      }
      if (change == 1)
         y += s2;
      else
         x += s1;
      weight += 2*dely;
   }
}

int sign(x)
int x;
{  int abs();
   if(abs(x) != x)
      return(-1);
   else if (x==0)
      return(0);
   else
      return(1);
}
```

รูปที่ 4 โปรแกรมลากเส้นตรงและวงกลมที่เขียนขึ้นจากอัลกอริทึมของ Bresenham

if $d_i > 0$ then 3

if $d_i = 0$ then 20

ตัดสินใจกรณีที่ 1 หรือ 2

2 $z = 2d_i + 2y_i - 1$

if $z < = 0$ then 10

if $z > = 0$ then 20

ตัดสินใจกรณีที่ 4 หรือ 5

3 $z' = 2d_i - 2x_i - 1$

if $z' < = 0$ then 20

if $z' > = 0$ then 30

10 $x_i = x_i + 1$

$d_i = d_i + 2x_i + 1$

go to 1

20 $x_i = x_i + 1$

$y_i = y_i - 1$

$d_i = d_i + 2x_i - 2y_i + 2$

go to 1

30 $y_i = y_i - 1$

$d_i = d_i - 2y_i + 1$

go to 1

4 finish

เมื่อนำ Bresenham's algorithm มา Implement ด้วยภาษา C ก็จะได้ฟังก์ชันเส้นตรงและวงกลมดังรูปที่ 4 (สำหรับวงกลมนั้น อัลกอริทึมเขียนไว้เฉพาะเสี้ยวหนึ่งเท่านั้น ในกรณีที่เราใช้เขียนทั้งวงต้อง Plot ทั้ง 4 Quadrant พร้อมๆ กัน)

มาถึงตอนนี้คุณก็จะได้กราฟิกไลบรารี่ไว้ใช้งานกับ BDS C คอมพายเลอร์แล้ว อย่างน้อยก็ 5 ฟังก์ชัน ส่วนโปรแกรมหลักนั้นผมเหลือให้คุณทำเป็นการบ้าน เขียนโปรแกรมเรียกใช้ฟังก์ชันกราฟิก วาดรูปสวยๆ ไว้ดูเล่นก็แล้วกัน ผมเองได้ลองคอมพายล์และเรียกใช้งานดูแล้วก็เห็นว่ายังช้าอยู่บ้างเล็กน้อย แต่ก็เร็วกว่าภาษาเบสิกมาก ถ้าคุณมี C คอมพายเลอร์ตัวอื่นก็ลองคอมพายล์ดูแล้วบอกต่อๆ กันบ้างนะครับว่าเป็นอย่างไรกันบ้าง โอกาสหน้าผมจะหาโปรแกรมภาษา C มาฝากอีก